Identifying, Classifying and Resolving Non-Sentential Utterances in Customer Support Systems

Poulami Debnath
Accenture Labs
Bangalore, India
poulami.debnath
@accenture.com

Shubhashis Sengupta
Accenture Labs
Bangalore, India
shubhashis.sengupta
@accenture.com

Harshawardhan M. Wabgaonkar
Accenture Labs
Bangalore, India
h.wabgaonkar
@accenture.com

Abstract

Task-oriented virtual agents (VAs) are expected to interact with human users in a natural language such as English and work with them to perform the users' desired tasks. In order to respond to the user or to carry out other actions, the VA needs to understand the meaning of the user utterances. Since humans often utter (syntactically) incomplete sentences during conversations, the VA needs to have the ability to comprehend such incomplete utterances - also known as Non Sentential Utterances (NSU). In this work, we propose algorithms for the detection, classification and resolution of such incomplete natural language utterances. Both rule-based as well as machine-learned algorithms are proposed for NSU detection. The NSU classification algorithm is machine-learning based. The output from the detection and classification tasks is used by a heuristic algorithm for the NSU resolution task. Experimentations on and results of these algorithms are presented and discussed for three different corpora (real-life human agent-user chats from hospitality, retail and information-technology support areas) related to Customer Support Representative domain.

1 Introduction

The onset of voice or chat-based assistants has created a need for building virtual agents that will be able to carry out more natural conversations with humans. More specifically, we examine the domain of Customer Support Representatives (CSR) that form an integral part of any business organization. Great customer service and engagement drive business growth and popularity as discussed in a survey (Sprinklr, 2017) and the use of chatbots or virtual agents is considered to be a way to achieve those positive business outcomes (Gautam, 2017). A virtual agent that understands human utterances even in their partial forms would make communication more natural.

The bot assistants available today (like Alexa or Siri) exhibit question-and-answering functionality, being pre-trained with commands (Martin and Priest, 2017) in certain areas. As such the voice command systems do not have session handling capability of their own (Dart, 2017) and external skill-sets have to be written to enable conversational capabilities. Handling all nuances of natural conversations, specifically the NSUs, is yet to be seen in these systems.

A: *as far as purchases, the sales department can help you purchase it. However as soon as you purchase it we can help you install it*
B: *excellent....*

Example 1: Retail Transcript Snippet

A: *We do also offer discounts for AAA members and seniors 62 and over based on availability. If you would qualify I can check for these rates as well.*
B: *AAA*

Example 2: Hospitality Transcript Snippet

Examples 1 and 2 are transcript snippets from the customer service domains of retail and hospitality, respectively. Speaker A depicts the human agent and B, the customer. In both examples, B's response is partially complete. The first instance is an exclamation by the customer whose intended expression is to be understood by the agent. The second example is a response to agent's context parameter (membership_type).

Handling NSUs is critical to deriving the full semantic meaning of the conversation and recent neural models such as sequence-to-sequence (Vinyals and Le, 2015) address only a few of these issues. We discuss some prior work in section 2. We attempt the detection of partial utterances using both rule-based as well as machine learning based approaches, followed by machine learning based approaches to classify partial utterances, as described in section 3. The results are discussed in section 4. We discuss our resolution approach in section 5 and conclude this paper in section 6.

2 Related Work

Non-sentential or *elided* utterances have been analyzed by researchers. A detection methodology for verb phrase ellipsis using machine learning was presented by (Nielsen, 2004). (Pulman, 2000) discusses a conditional equivalence mechanism of resolution between quasi-logic forms and their resolved logic forms that cater to verb phrase elliptical occurrences. (Hardt and Rambow, 2001) examine the factors for eliding verb phrases in text and present a trainable model.

(Fernández and Ginzburg, 2002) conducted a corpus-based study on some transcripts from the British National Corpus (BNC) and presented a taxonomy of NSUs. A particular class- *sluice*, was extensively studied by (Fernández et al., 2004). A machine learning classification for NSU types in dialog was conducted by (Fernández et al., 2005) for the BNC corpus.

(Dragone, 2015) built on the classification work of (Fernández, 2006) by incorporating additional features and a semi-supervised learning technique which resulted in an improvement in the classification accuracy and also provided an approach to the probabilistic modeling of the dialog context. The author reformulated the incomplete-sentence resolution rules from (Fernández et al., 2005) with a probabilistic account of the dialog state.

(Schlangen, 2005) presented a machine learning based approach to identify fragmentary sentences and their antecedents in a multi-party dialog. (Lin et al., 2016) presented an ellipsis and coreference module in a virtual patient dialog system. (Raghu et al., 2015) proposed a rule-based approach to generate resolved questions based on an input corpus of template reference questions and ranking the results. Another approach was taken by (Kumar and Joshi, 2016) using an RNN based encoder-decoder network, that would create the resolved utterance based on the incomplete utterance and its dialog context. They trained sequence models for semantic as well as for syntactic patterns followed by building an ensemble model.

3 Our Approach

3.1 Overall Architecture of the Spoken Dialog System

Figure 1 shows the overall architecture of our prototypical spoken dialog system, built mainly over open source software. When a customer (Cust) utterance (*utt*) is received by the system, the constituent blocks (Incoming Utterance Analyzer, Spoken Language Understanding Unit, Dialog Manager, Natural Language Generator, Response Interface unit) work together to generate a response (*resp*) to *utt*. In this paper, we only discuss the working of the Partial Utterance Analyzer (PUA) which is a sub-block of the Spoken Language Understanding unit. The dotted box on the left-hand side of Figure 1 shows the flow of the conversation timeline. *ant* refers to the antecedent, the immediately previous sentence uttered by the system. The PUA is composed of three modules - the Detector, the Classifier and the Resolver modules.

3.2 Assumptions

We have made the following assumptions. First, the virtual agent would always converse in complete sentences. Thus, we focus on resolving human partial utterances only. Second, the underlying methodology of this partial utterance analyzer module is meant to be applied to only dialog scenarios. These have not been tested on other forms of text- like essays, interviews, or multi-party conversations.

3.3 Corpora: Corpus I for Detection and Corpus II for Classification

We have curated two sets of corpora of user utterances along with their immediate antecedent texts. The first corpus, Corpus I, of size 1497 is used only for detection experiments. It contains both positive (637)

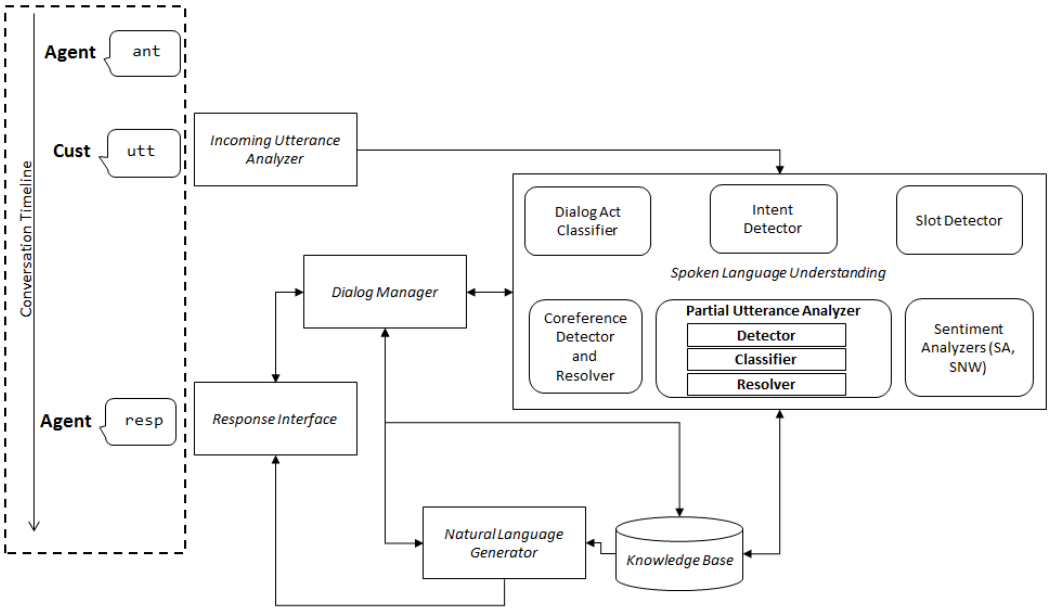

Figure 1: Overall Architecture

and negative (860) occurrences of non-sentential user utterances. Our annotations of the target value is either a *yes / no*, indicating the presence / absence of an NSU. The second corpus, Corpus II, of size 900 is used to train machine learning models for the classification of NSU categories. All user utterances in this corpus, are therefore, of the non-sentential type.

For both corpora, we have taken real-life chat transcripts across the industry domains of hospitality, retail and IT support. Hospitality transcripts include discussions around room booking, user profile related issues and customer-rewards/offers. Chats in retail are mainly around product usage support, replacement of defective items, and troubleshooting procedures. IT chat transcripts comprise discussions around technical assistance, network troubleshooting and so on.

3.4 Advice Codes

We pre-define a catalog of advisory codes that give execution instructions to handle the incomplete utterance. Our rule based detection algorithm additionally produces a response code along with the expected yes/no evaluation, based on the rule that was triggered to arrive at the detection outcome. The classifier module associates a specific advice code with each of the NSU classes. These advisory codes are used by the Resolver module to understand the partial utterances and enable the VA to move the conversation forward.

3.5 The Detector Module

We discuss both the rule-based and the machine-learning based approaches on Corpus I with 1497 records.

Rule Based Detection of Non-Sentential Utterances

The rule-based methodology presented in Algorithm 1 takes the inputs: *utt* (user utterance), *ant* (antecedent) and *dac_utt* (dialog act class [1] of *utt*). The detection outcome evaluated using a rule-engine could be one of the following: *yes* (*utt* is non-sentential), *no* (*utt* is complete) or *rule not found* (*utt* could not be covered by any of the rules). We've grouped the detection rules into multiple subroutines

[1]The Dialog Act Classifier (DAC) is one of the modules of our Spoken Language Understanding block and it predicts one of the following acts for an utterance: G_G (greeting), CAT (Confirmation Affirmation Turn), INFORMATION, COMMAND or QUESTION. Other details related to DAC are out of the scope of this paper.

as shown in algorithms 1 - 9. *aff*, *conn*, *greet*, *rej*, *sluice* are pre-defined sets of affirmation words, connective words, greeting words, rejection words and question words, respectively. The flag variables *fr_beg* and *fr_end* indicate the presence of connective words at beginning and at the end of utterance, respectively. *av_standalone* indicates if an auxiliary verb has an associated main verb in *utt*. *mv*, *nc* and *noun_pos* respectively denote the extracted main verbs, noun chunks and noun pos tags from *utt*. *SVO_structure* represents the structure of *utt*, it could be VO (verb object), SV_simple (subject verb in a simple sentence) or SV_compound (subject-verb in a compound sentence) or SVO (subject-verb-object). Finally, *tkn* contains the word token count of *utt*. We formulated these rules by first analyzing the various NSU utterances, starting with smaller utterances (single token) and iteratively visiting larger sentences and also studying their dependency trees.

Machine Learning Based Detection of Non-Sentential Utterances

The machine learning based approach uses scikit's (Pedregosa et al., 2011) Support Vector Machines for model training with a linear kernel parameter. Given the user utterance, its antecedent and both their dialog act classes, the feature computation process is automated. One of the features is the length type of the utterance which could be either *single* (single token utterance), *small* (smaller number of tokens, up to four tokens) or *long* (more than four tokens). Another feature is the type of the utterance structure in terms of its subject-verb-object (SVO) components. The utterance's PoS (Parts-of-speech) uni-grams and bi-grams are used as another feature. The first two features are categorical, therefore we encode them using LabelEncoder and OneHotEncoder. The third feature is of string type to which we apply CountVectorizer. The train/test split ratio used is 80/20 for this experiment.

3.6 The Classifier Module

The classifier module's function is to predict the class type of an NSU. The categorization of NSU classes has been motivated from the taxonomy of (Fernández and Ginzburg, 2002). We analyzed the customer service chat transcripts and based on the nature of our domain, merged a few of these into a single class. On the other hand, we have ignored a few of the non-relevant ones. We have also added the classes Verb Phrase Ellipsis (VPE) and Noun Phrase Ellipsis (NPE). We have used nine classes of NSUs in our work: Ack (Acknowledgement), AffAns (Affirmation Answer), FragByConn (Fragments By Connectives), NPE, PropModifier (Propositional Modifier), RejAns (Rejection Answer), Short Answer, Sluice and VPE. Table 1 shows the description of these classes with examples where B's utterances are non-sentential (shown in italics). Specific advice codes are mapped to each of these individual classes that later help with resolution.

As stated earlier, Corpus II is used for the classification experiments. The features used are listed in Table 2a. Table 2b shows the distribution of the NSU classes in this corpus. All the features are computed automatically out of which one is of string type and the rest are categorical. Encoders LabelEncoder and OneHotEncoder are used for the latter type whereas uni-grams and bi-grams are computed for the individual tokens and their corresponding PoS tags. The dataset is split into 80/20 train/test ratio. We have trained two models– Support Vector Machines on a linear kernel and Random Forests with 4000 trees using scikit-learn (Pedregosa et al., 2011), on this data and using these features.

4 Results

4.1 Detection Results

Table 3a summarizes the results of rule-based detection algorithm on Corpus I under column name *Combined*. For comparing the efficiency of the detection algorithm across domains, we also present the individual results for *Hospitality*, *Retail* and *IT* support. We observe that the coverage of these rules is quite high, at least 91%. The accuracy measures around 88%, with the precision, recall and f1 scores hovering around 0.85 for the combined dataset. Table 3b shows the metrics of the ML based detection approach. The average precision, recall and f1 scores are 0.82 which are slightly lesser than the rule-based approach.

Input: (utt, ant, dac_utt)
Output: detection_output
Result: Partial Utterance Detection Outcome
res = Call Sub_1 ()
if *res != empty* **then**
  return *res*
if *tkn == 1* **then**
  Call Sub_2 ()
if *tkn in {2,3,4} and nc == empty and mv == empty* **then**
  Call Sub_3 ()
if *tkn >1 and mv is empty and noun_pos is empty* **then**
  if *regex((PROPN(CCONJ(PROPN))*)+) == True or regex((((NUM)*)((PROPN)*))+) == True* **then**
    return *yes*
if *tkn >4 or (tkn in {2,3,4} and either nc or mv is non-empty)* **then**
  Sub_4 ()
if *utt contains sluice text and verb is missing* **then**
  return *yes*
if *utt has verb missing* **then**
  return *yes*
return *"rule not found"*

Algorithm 1: Detection Algorithm

Input: (utt)
Output: detection_output
if *utt in greet and utt in {aff, rej}* **then**
  return *yes*
if *utt in conn and (fr_beg == True or fr_end == True)* **then**
  return *yes*
if *utt in greet* **then**
  return *no*
if *utt has av_standalone* **then**
  return *yes*

Algorithm 2: Sub_1

Input: (utt, dac_utt)
Output: detection_output
if *utt in {aff, rej, sluice}* **then**
  return *yes*
if *utt in greet* **then**
  return *no*
if *dac_utt == "CAT"* **then**
  return *yes*
if *pos_utt in {"ADV", "ADP", "PROPN", "NOUN", "INTJ", "VERB", "NOUN"}* **then**
  return *yes*

Algorithm 3: Sub_2

Input: (utt, dac_utt)
Output: detection_output
if *utt in greet* **then**
  return *no*
if *utt in {aff, rej, sluice}* **then**
  return *yes*
if *dac_utt == "CAT"* **then**
  return *yes*

Algorithm 4: Sub_3

Input: (utt)
Output: detection_output
if *utt has VO* **then**
  Sub_4.1 ()
if *utt has SV_simple* **then**
  Sub_4.2 ()
if *utt has SV_compound* **then**
  Sub_4.3 ()
if *utt has SVO* **then**
  Sub_4.4 ()

Algorithm 5: Sub_4

Input: (utt)
Output: detection_output
if *firstword(utt) in {aff, rej}* **then**
  return *yes*
if *dac_utt == "COMMAND"* **then**
  return *no*
else if *dac_utt == "QUESTION"* **then**
  return *yes*

Algorithm 6: Sub_4.1

Input: (utt)
Output: detection_output
if *firstword(utt) in {aff, rej}* **then**
  return *yes*
if *helping_verb(utt) == True* **then**
  return *yes*
if *helping_verb(utt) == False and (acomp(verb,ADJ) == True or advmod(verb,ADV) == True)* **then**
  return *no*
if *helping_verb(utt) == False and xcomp(verb,-) == True* **then**
  return *yes*

Algorithm 7: Sub_4.2

Input: (utt)
Output: detection_output
if *firstword(utt) in {aff, rej}* **then**
  return *yes*
if *(nsubj(verb1,-) == True or nsubjpass(verb1,-) == True) and acomp(verb2, ADJ)* **then**
  return *no*
if *(nsubj(verb1,-) == True or nsubjpass(verb1,-) == True) and advmod(verb2, ADV)* **then**
  return *yes*
if *(nsubj(verb1,-) == True or nsubjpass(verb1,-) == True) and attr(verb2, nounphrase)* **then**
  return *no*

Algorithm 8: Sub_4.3

Input: (utt)
Output: detection_output
if *firstword(utt) in {aff, rej}* **then**
  return *yes*
else
  return *no*

Algorithm 9: Sub_4.4

We have adopted the rule-based detection in our prototype implementation because of two main reasons: Advisory codes produced by rule algorithm give additional information that which rule was triggered to arrive at the conclusion, this is not available in the ML approach. For example, information about the occurrence of fragment words at either the beginning or at the end of an utterance could be obtained by rules; while the ML approach only yields a binary outcome (yes / no). This information becomes valuable during resolution. Second, the resultant metrics are a little better for the rule approach

than that of the ML one. An auxiliary benefit of using rules is that they give an insight into the coverage efficiency of the rule-set. This could further help us in identifying linguistic heuristics to improve coverage, such as adding domain-specific data matches for ticket reference numbers, log data, etc.

NSU Class Name	Description	Example
Ack (Acknowledgement)	user acknowledgement to antecedent	A: I will run a scan for errors. B: *ok*
AffAns (Affirmation Answer)	user confirming acceptance of antecedent	A: Would you like to get us started? B: *yes please*
FragByConn (Fragments by Connectives)	usage of fragments (*but*, *and*, *or*, *as well as*) at the beginning or at the end of *utt*	A: Please try to enter your password into the other box. B: *I don't know and*
NPE (Noun Phrase Ellipsis)	noun part being omitted in *utt*	A: What specific symptoms are you having? B: *breaks up*
PropModifier (Propositional Modifier)	exclamations using adjectives, adverbial words	A: Average wait time is 2-32 minutes B: *excellent*
RejAns (Rejection Answer)	*utt* expressing rejection of antecedent	A: Is the forecast lost? B: *no*
Short Answer	*utt* containing just the answer values	A: What type of computer do you currently have? B: *Microsoft Surface Pro 4*
Sluice	questions in incomplete forms	A: Can you clear your cache? B: *how?*
VPE (Verb Phrase Ellipsis)	verb part being omitted in *utt*	A: Have you tried using another browser like Google Chrome to do the printout?? B: *no I haven't*

Table 1: NSU class description with examples from customer chat transcripts (A: Agent, B: Customer)

Feature	Description
wh	presence of wh-word in utt
aff	presence of affirmation word in utt
rej	presence of rejection word in utt
ack	presence of acknowledgement in utt
frag	presence of fragment words in utt
grams	pos and word n-grams of utt, n in (1,2)
utt_dac	DAC class of utterance
ant_dac	DAC class of antecedent
len	if length of utterance is single, short or long
svo	subject-verb-object structure of utt
noun	presence of noun in utt
mv	presence of main verb in utt
av_mv	auxiliary verb in utt having an associated main verb in utt
single_token_type	type of utt if it consists of only one word
firstword	type of first word in utt

(a) Feature Set

Class Type	Count
Ack	93
AffAns	140
FragByConn	79
NPE	87
PropModifier	41
RejAns	77
ShortAnswer	210
Sluice	56
VPE	117
Total	900

(b) Class Distribution

Table 2: Set of features and class distribution of the classification corpus

4.2 Results of Partial Utterance Classification

Tables 4a and 4b show the classification reports of the models trained using SVM and using RF. Both show similar average results for the precision, recall and f1 parameters. Short Answer type that had maximum count in class distribution showed good recall values in SVM (0.93) and RF (0.88) models. All classes except NPE and VPE show good precision values in SVM. The limited and similar type of data points of NPE could be the reason for its bad performance. The Random Forest classifier shows high precision for Sluices (1.00) and Affirmative Answers (0.96). Please note that the corpora references can be given upon request.

Parameter	Hospitality	Retail	IT	Combined
Dataset Size	493	498	506	1497
Coverage	96.35%	91.77%	92.29	93.45%
Accuracy	89.68%	87.75%	86.94%	88.06%
Precision	0.8457	0.8778	0.8028	0.8417
Recall	0.8509	0.8700	0.9067	0.8752
f1	0.8483	0.8739	0.8516	0.8581

(a) Rule-Based Detection Approach

	Precision	Recall	F1
no	0.85	0.85	0.85
yes	0.77	0.77	0.77
avg / total	**0.82**	**0.82**	**0.82**

(b) Machine Learning based Detection Approach using Support Vector Machines

Table 3: Results of Partial Utterance Detection

5 The Resolver Module

5.1 Defining Resolution in Customer-Support Chat Scenarios

The aim of resolving an NSU in a goal-oriented conversation is to enable the agent to understand its intended meaning and progress the conversation accordingly. It may not always necessarily mean reconstructing the utterance. With this understanding, we've designed the resolver module to interact with some of the other components of our system- namely the dialog manager that helps with a meaningful conversation flow; it also consists of a dialog state tracker (keeps track of state variables in a conversation), and a policy manager (decides on strategies like grounding, confirmation questions, taking turns). As discussed earlier, outcomes from the rule-based detector and classifier modules guide the resolver to handle the partial utterance.

	Precision	Recall	F1
Ack	0.91	0.91	0.91
AffAns	0.85	0.81	0.83
FragByConn	0.73	0.62	0.67
NPE	0.55	0.69	0.61
PropModifier	0.82	0.64	0.72
RejAns	0.82	0.93	0.87
ShortAnswer	0.89	0.93	0.91
Sluice	1.00	0.70	0.82
VPE	0.67	0.70	0.68
avg / total	**0.81**	**0.81**	**0.81**

(a) Classification Results for SVM

	Precision	Recall	F1
Ack	0.87	0.91	0.89
AffAns	0.96	0.85	0.90
FragByConn	0.69	0.69	0.69
NPE	0.64	0.88	0.74
PropModifier	0.69	0.64	0.67
RejAns	0.76	0.87	0.81
ShortAnswer	0.83	0.88	0.85
Sluice	1.00	0.60	0.75
VPE	0.75	0.60	0.67
avg / total	**0.81**	**0.80**	**0.80**

(b) Classification Results for Random Forests

Table 4: Classification Results

5.2 The *resolve* Function

We formulate the following function in order to resolve non-sentential utterances in a goal-oriented conversation:

$$resolve(getCODE(obj), getCTXT(obj.utt), getCTXT(obj.ant), statevar, turn\_flag) \quad (1)$$

A dialog object *obj* includes the user NSU (*obj.utt*) and its antecedent (*obj.ant*). The *resolve* function consists of the sub-functions *getCODE()* and *getCTXT()*, and parameters *statevar* and *turn_flag*. *getCODE()* retrieves and merges the advisory codes from the detector and the classifier modules. *getCTXT()* retrieves the context variables at the current dialog level. These may include specific slot values or even actionable items. The context information is retrieved for both *obj.utt* and *obj.ant*, as shown by the second and third parameters of the *resolve* function. *statevar* is the set of all state variables at the entire conversation level. *turn_flag* indicates the next turn taker- 0 implies system's turn, 1 implies user's turn. We describe the resolution steps through algorithms 10 through 12. The output of the *resolve* function is a series of suggested execution steps based on the advisory code.

Here, we show the *code* values as the NSU class names. Some supplementary systems are used, e.g. sentiment analyzer, question answering(Gupta et al., 2018). We associate *conf flags* (confirmation flags)

for all variables and a confirmation by the user sets the associated flags to 1. The function *isAdditional-Text()* (Algorithm 11) takes checks if there is additional text present in the utterance *text* other than the pre-defined sets of *Ack/ AffAns/ RejAns*. Algorithm 12 sets the value of turn_flag.

```
Input: resolve(code, obj.utt, obj.ant, statevar, turn_flag)
Output: resolution steps based on NSU class code
if code in "Ack", "AffAns", "RejAns" then
    if code is "Ack" or "AffAns" then
        set conf flags to 1
    else
        set conf flags to 0
    if isAdditionalText(code, obj.utt) == yes then
        update context vars in obj
if code == "PropModifier" then
    Compute Sentiment and Emotion scores
    Invoke Policy Mgr to generate apt response
if code == "Short Answer" then
    Assign context var of obj.ant with obj.utt
if code == "Sluice" then
    Invoke Question Answering system to get answer
if code in "FragByConn_beg", "FragByConn_End then
    if "beg" in code then
        reconstruct obj.utt by appending obj.utt to obj.ant
    if "end" in code then
        wait for user to enter further input text
if code == "NPE" then
    retrieve noun phrases from obj.ant
    ask user confirmation based on policy manager
if code == "VPE" then
    retrieve action verbs from obj.ant
    ask user confirmation based on policy manager
update statevar; set_turn_flag(code)
```

Algorithm 10: Resolution

```
Input: isAdditionalText(code, text)
Output: Checking for additional data
if code == "Ack" then
    Check token_count in text after
      removing words from Ack set
if code == "AffAns" then
    Check token_count in text after
      removing words from Aff set
if code == "RejAns" then
    Check token_count in text after
      removing words from Rej set
if token_count == 0 then
    return no
else
    return yes
```

Algorithm 11: Checking Additional Text

```
Input: set_turn_flag (code)
Output: Setting turn_flag
if code in "Ack", "PropModifier"
  then
    turn_flag = 0
if code in "FragByConn_End" then
    turn_flag = 1
```

Algorithm 12: Setting turn_flag

User's response *excellent* in example 1 is a *PropModifier*, the resolver would check the sentiment and emotion scores and invoke the policy manager to generate a response. In example 2, user utterance is *Short Answer* that would update the antecedent context variable membership_type.

6 Conclusion

We present a Partial Utterance Analyzer that detects, classifies and resolves non-sentential utterances for human-BOT conversations for customer services. We discuss a rule-based and a machine learning approach for detection. For classification, we show machine learning models. Resolution involves executing instructions from advices codes that are generated by the rule-based detection and the classification modules. Results of detection and classification are fairly good, considering the open-ended and practical nature of the data. The corpora have been curated from real-life chat transcripts across hospitality, retail and information technology support areas.

There isn't a way of directly comparing our work with those of the earlier approaches, primarily because of the nature of the data (real-life chats) and the nature of the domain (goal-oriented customer service chats). There are no corpus-specific constructs (e.g. *has_pause*) or embedded data (like C5 tags) as were there in the BNC corpus. We have refurbished the set of NSU class types from what is described in earlier work by merging some of the classes, adding new ones and leaving out a few that don't seem to attach any relevance in a chat-bot framework. Our resolution approach is tied to advice codes that the dialog manager architecture supports with its functionality, whereas earlier approaches were mainly around reconstructing sentences, thus making comparison a tricky process.

Our current work is integrated in a prototypical framework called OpenDial (Lison, 2015), which is a Java toolkit for developing spoken dialog systems using a probabilistic-rules formalism. As we deploy our framework in practice, our future work will focus on a detailed analysis of the performance of the algorithms by testing them with more data across CSR domains as well as on social chat-bot scenario, and on improving the robustness of the algorithms. We also wish to explore the transitioning towards automatic rule induction, and inclusion of deep learning techniques at various stages of the algorithms.

References

Scott Dart. 2017. Tips on state management at three different levels @MISC, May.

Paolo Dragone. 2015. Non-sentential utterances in dialogue: Experiments in classification and interpretation. *CoRR*, abs/1511.06995.

Raquel Fernández and Jonathan Ginzburg. 2002. Non-sentential utterances in dialogue: A corpus-based study. In *Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue - Volume 2*, SIGDIAL '02, pages 15–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2004. Classifying ellipsis in dialogue: A machine learning approach. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Raquel Fernández, Jonathan Ginzburg, and Shalom Lappin. 2005. Using machine learning for non-sentential utterance classification.

Raquel Fernández. 2006. *Non-sentential Utterances in Dialogue: Classification, Resolution and Use*. Ph.D. thesis, King's College London.

Nitish Gautam. 2017. Customer service chatbot - using chatbots for customer service and customer support @MISC, October.

Deepak Gupta, Rajkumar Pujari, Asif Ekbal, Pushpak Bhattacharyya, Anutosh Maitra, Tom Jain, and Shubhashis Sengupta. 2018. Can taxonomy help? improving semantic question matching using question taxonomy. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 499–513. Association for Computational Linguistics.

Daniel Hardt and Owen Rambow. 2001. Generation of vp ellipsis: A corpus-based approach. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 290–297, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vineet Kumar and Sachindra Joshi. 2016. Non-sentential question resolution using sequence to sequence learning. In Nicoletta Calzolari, Yuji Matsumoto, and Rashmi Prasad, editors, *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 2022–2031. ACL.

Chuan-Jie Lin, Chien-Wei Pao, Yen-Heng Chen, Chi-Ting Liu, and Hui-Huang Hsu. 2016. Ellipsis and coreference resolution in a computerized virtual patient dialogue system. 40, 09.

Pierre Lison. 2015. Developing spoken dialogue systems with the opendial toolkit. In *Proceedings of the 19th Workshop on the Semantics and Pragmatics of Dialogue. Goteborg*.

Taylor Martin and David Priest. 2017. The complete list of alexa commands so far @MISC, December.

Leif Arda Nielsen. 2004. Verb phrase ellipsis detection using automatically parsed text. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Stephen G. Pulman. 2000. Bidirectional contextual resolution. *Comput. Linguist.*, 26(4):497–537, December.

Dinesh Raghu, Sathish Indurthi, Jitendra Ajmera, and Sachindra Joshi. 2015. A statistical approach for non-sentential utterance resolution for interactive QA system. In *Proceedings of the SIGDIAL 2015 Conference, The 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2-4 September 2015, Prague, Czech Republic*, pages 335–343. The Association for Computer Linguistics.

David Schlangen. 2005. Towards finding and fixing fragments: Using ml to identify non-sentential utterances and their antecedents in multi-party dialogue. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 247–254, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sprinklr. 2017. 4 lessons for delivering effective social customer care @MISC.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *CoRR*, abs/1506.05869.